is--;

Page 15, line 4, delete "this" and substitute –the--;

Page 15, after line 25, insert the following new paragraph:

A6

--While this invention has been described in conjunction with specific embodiments thereof, it is evident that many alternatives, modifications and variations will be apparent to those skilled in the art. Accordingly, the preferred embodiments of the invention as set forth herein, are intended to be illustrative, not limiting. Various changes may be made without departing from the true spirit and full scope of the invention as set forth herein and defined in the claims.—

IN THE CLAIMS:

Please cancel claims 1 – 5 in their entirety and without prejudice and substitute the following new claims.

A7

--6. A method for verifying transformation of a source code (1) into a transformed code (3) designed for an embedded system, said source and transformed codes being associated with virtual machines, characterized in that it comprises at least the following steps:

- determining, for each of said source (1) and transformed (3) codes, a first common subset (13), constituting a single virtual machine that factors in the behavior of said source and transformed codes (1,3);

- determining, for each of said source (1) and transformed (3) codes, a second subset (10, 30) constituted by a plurality of so-called auxiliary functions ($10_i - 30_i$) used by said single virtual machine, said auxiliary functions ($10_i - 30_i$) representing residual differences between said source (1) and transformed (3) codes;

- associating said auxiliary functions in pairs, a first auxiliary function (10_i) of each pair belonging to said second subset (10) associated with said source code (1) and a second auxiliary function (30_i) of each pair belonging to said second subset (30) associated with said transformed code (3);

- verifying (6) a given correspondence property between said auxiliary functions ($10_i - 30_i$) of all of said pairs; and

- verifying that said transformation of the source code (1) into a transformed code (3) satisfies said given correspondence property.

7. A method according to claim 6, characterized in that said correspondence property is a logical relation, so that said auxiliary functions of each of said pairs (10_i - 30_i), when executed, generate results linked by said logical relation.

8. A method according to claim 6, characterized in that said logical relation is an identity relation for observable entities of each of said source and transformed codes, for any pair of auxiliary functions, so that the functionalities of said source code (1) are retained when said transformation into said transformed code (3), and said verification of the code transformation are performed.

9. A method according to claim 6 further comprising applying the steps of the verification of transformation to a code transformer (2) and generating from said source code (1), a transformed code (3) in a memory (71) of a chip card (7).

10. A method according to claim 7 further comprising applying the steps of the verification of transformation to a code transformer (2) and generating from said source code (1), a transformed code (3) in a memory (71) of a chip card (7).

11. A method according to claim 9, characterized in that, said transformed code is a program written in the virtual machine of a given

computer language, and said chip card (7) stores a plurality of software applications (A_1 through A_n) written in said transformed code (3).

12. A method according to claim 10, characterized in that, said transformed code is a program written in the virtual machine of a given computer language, and said chip card (7) stores a plurality of software applications (A_1 through A_n) written in said transformed code (3).

13. A method according to claim 9, characterized in that said source code (1) is a program written in a "JAVA" virtual machine and said transformed code (3) is a program written in a "JAVA CARD" virtual machine.

14. A method according to claim 10, characterized in that said source code (1) is a program written in a "JAVA" virtual machine and said transformed code (3) is a program written in a "JAVA CARD" virtual machine.

15. A method according to claim 11, characterized in that said source code (1) is a program written in a "JAVA" virtual machine and said transformed code (3) is a program written in a "JAVA CARD" virtual machine.

16. A method according to claim 12, characterized in that said source code (1) is a program written in a "JAVA" virtual machine and said transformed code (3) is a program written in a "JAVA CARD" virtual machine.--